Gene Expression Analysis on BRCA1-Deficient Mouse Tumors

Shalvi Chirmade

December 17, 2021

GitHub link: https://github.com/shalvichirmade/BRCA1-Deficient-Mouse-Tumors-Gene-Expression-Analysis

Introduction

I began my interest in genetic counselling when I was in high school. This came from a small paragraph in my biology textbook and evolved into a career passion. As you may know, genetic counselors are health professionals that help guide patients with understanding and providing information on their genetic reports and/or disorders. This can range from fields of early developmental and birth defects to inherited disorders and familial cancer. I personally have volunteered and worked with genetic counselors after graduating from my BSc and continued to be entranced by the work and research they conduct. My goal was to one-day become a genetic counselor myself until I found the field of bioinformatics and realized I could turn my fascination of human genetic disease into computational analysis. This is the main reason I decided to search for a data set analyzing breast cancer tumors. A comprehensive understanding of how BRCA1 affects the formation of breast and/or ovarian tumors is not complete so research of this kind is important for society. It aids in acquiring knowledge of hereditary and sporadic carcinomas (Yoshida & Miki, 2004).

BRCA1 is an important gene responsible for DNA repair, cell cycle checkpoints and response to DNA damage (Yoshida & Miki, 2004). It also plays a crucial role in genomic stability; the protein synthesized interacts with its proteome to allow for recognition and repairing of damaged DNA (Jhanwar-Uniyal, 2003). As it interacts with multiple different proteins to carry out this process, it plays a very important role in suppressing tumorigenesis (Jhanwar-Uniyal, 2003). BRCA1 is a well-conserved gene among various species, conducting research in hopes to understand its biological process is crucial for humans and other related organisms (Jhanwar-Uniyal, 2003). Some mutations in BRCA1 mask the functional ability of the wild-type allele which causes a higher probability of the individual developing a breast cancer (Jhanwar-Uniyal, 2003). Mutations of tumor-inducing genes, such as BRCA1, allow for an early-onset of breast and/or ovarian cancer in women (Sun *et al.*, 2021). The research conducted by these authors investigate the origin of mammary tumors from the luminal epithelia and compare the gene expression to cells derived from the mammary tumors themselves. The data set derived from this study is what I will be using for this script; it will be described in greater detail in the next section. As for the gene expression analysis being conducted in this script, choosing the correct software tools for statistical analysis is of great importance. I will go into deeper detail about this later on when explaining which software tool I decided to use.

The question I would like evaluate for this script is, what are the biological processes of the differentially expressed (DE) genes in this data set? Are the DE genes related to the biological function of BRCA1, i.e., are they part of the individual networks for DNA repair and damage or are their functions on a broader spectrum like immune response or apoptosis? I plan to carry out gene expression analysis using the data set obtained by Sun *et al.* by utilizing the packages limma and goseq by Bioconductor.

Data Set

I was searching through the GEO database when I came across this data set (November 22, 2021). It comes from a very interesting paper written by Sun *et al.* in 2021 titled "Dissecting the heterogeneity and tumorigenesis of BRCA1-deficient mammary tumors via single cell RNA sequencing". This data set was obtained and used by the authors for expression profiling and the understanding of the BRACA1-deficient mouse tumors via RNA sequencing. BRCA1 is a gene known to be tumor-inducing when mutated (Sun et. al., 2021); this can occur either sporadically or be inherited from a parent. This mutation can predispose breast and/or ovarian cancers (Yoshida & Miki, 2004). As the normal function of this gene is involved with DNA repair and apoptosis (Yoshida & Miki, 2004), we can understand how dysfunctional and detrimental a mutation in these regulatory processes can be. BRCA1 is aided by many regulatory proteins and it will be interesting to see if this data set shows differential expression in genes that interact with BRCA1. As all the mice utilized in this data set were BRCA1-deficient (the same cross, strain: C57BL/6 and 129/Sv mixed), the analysis carried out by the authors were on the differential expression between the luminal and tumor cells from their mammary glands. The authors main analysis was to showcase the heterogeneity of these cells in both intra- and inter-tumor levels. They wanted to assess the initiation and progression of tumor formation hence isolating samples from the luminal cells of the mammary gland and the tumor cells from the carcinoma of the same region. The data available on GEO has 477 samples which encompasses samples from eight different mice. Each of these mice are replicates from the same cross I mentioned earlier. There are between 30-60 samples from each mouse which I will discuss in the next section when carrying out my data exploration. As you will see, I found understanding this data set quite difficult. In the published paper, the authors do not correspond the names of the mice chosen to their publicly available data set hence the exact difference between each mouse was very hard to comprehend. The only difference I was able to find, once I went through about 100 of the 477 samples, was only if the sample came from luminal or tumor regions. As the authors also take into account gestation of the mouse, this information was not differentiated in the public data set. For each sample, there is a separate accession display but the description are all the same, which states "Three-to-four month-old female virgin or pregnant at day 12.5 mice were sacrificed". Hence, I was unaware if the mouse was either virgin or 12.5 days pregnant. This confusion carries on during the rest of my analysis as we will see that the differentially expressed genes are very minimal and not highly categorized. My speculation is, due to the fact that I am not explicitly aware of the difference between each sample, I may have grouped the samples in a less efficient way than they were intended. I have only differentiated the samples by luminal or tumor and have not taken into account gestation. I will discuss this further in my final results based on the analysis I accomplish.

The data set can be found at: https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE148569

Data Acquisition, Exploration, Filtering and Quality Control

Install Required Packages

```
#Bioconductor packages
#if (!requireNamespace("BiocManager", quietly = TRUE))
#install.packages("BiocManager")
#BiocManager::install(c("limma", "Glimma", "edgeR", "goseq", "Mus.musculus",
#                       "TxDb.Mmusculus.UCSC.mm39.refGene"))
library(edgeR)
library(Glimma)
library(goseq)
library(limma)
library(Mus.musculus)
library(TxDb.Mmusculus.UCSC.mm39.refGene)
```

```
#CRAN packages
#install.packages("gplots")
library(gplots)
#install.packages("RColorBrewer")
library(RColorBrewer)
#install.packages("R.utils")
library(R.utils)
#install.packages("tidyverse")
library(tidyverse)
```

Inputting Data

Reading in the data from the GEO database. As I am working on a Mac, it automatically unzips the file hence the file used here is only .txt. This file is provided to you.

```
#Load the complete data set into the script.
dfComplete_Data <- read.delim("GSE148569_scRNAseq_C1_count.txt")

#Inspect the data.
head(dfComplete_Data, n = c(10, 5))
```

```
##           X SC1_01 SC1_10 SC1_11 SC1_12
## 1     Xkr4      0      0      0      0
## 2      Rp1      0      0      0      0
## 3    Sox17      0      0      0      0
## 4   Mrpl15     13    224     36    125
## 5   Lypla1     25     23      0    151
## 6    Tcea1     19     93     43     24
## 7    Rgs20      0     22      0      0
## 8  Atp6v1h     82    273    612    218
## 9    Oprk1      0      0      0      0
## 10  Npbwr1      0      0      0      0
```

There are 477 samples in this data set and 21004 genes analyzed.

This data set has an association to a published paper by Sun *et al.* This paper is not explicit with the details of this data file so I found it quite difficult to come up with a conclusion for what the data is showing me. I speculate that each of the different SC samples correspond to a different mouse; so SC1, SC2 and so on depict the names of the particular mice. Hence, each sample associated with a particular SC is a sample from that mouse. For example, if we take the first sample from the data frame we visualized above, SC1_01 refers to the 01 sample from mouse SC1. So, sample SC5_29 would mean sample 29 from mouse SC5. Four of these mice have been used for tumor samples and the other four mice have been used for the luminal samples.

After talking with Sally about this data set, we came to a conclusion that the best approach for this assignment would be to randomly select three samples from each mouse for my further analysis. The next section of the code depicts this.

```
#I am going to duplicate the data frame to subset my data.
dfData <- dfComplete_Data
#Check dimensions.
dim(dfData)
```

```
## [1] 21004   478
```

```
#The first step is to make the columns displaying the gene names into the row names.
rownames(dfData) <- dfData$X
#Check to see if it worked
head(rownames(dfData),10)
```

```
##  [1] "Xkr4"    "Rp1"     "Sox17"   "Mrpl15"  "Lypla1"  "Tcea1"   "Rgs20"
##  [8] "Atp6v1h" "Oprk1"   "Npbwr1"
```

```
#Delete the gene name column.
dfData <- dfData[-1]
#Check to see if it worked.
dim(dfData)
```

```
## [1] 21004   477
```

Subset Data Frame

```
#Create a variable with the different SC names.
SC_Names <- c("SC1_", "SC2_", "SC4_", "SC5_", "SC6_", "SC7_", "SC9_", "SC10_")
```

Create a function to randomly select three samples from each mouse. As I have to do this eight times, I thought creating a function would be beneficial. This function can be used for any data frame, for any string to be matched and any number of samples. It will output a new data frame with the name of the string entered in the function.

```
Randomly_Sample_Columns <- function(df, str, n){

  #df - the name of the data frame from which you would like to sample columns

  #str - the string that you want to search for in the column names. Entered with ""

  #n - number of columns with the string you would like to randomly sample

  #Extract the string to use as the name of the data frame to be created
  name <- paste("df", str, sep = "")

  dfOutput <- df %>%
    select(matches(str)) %>%
    sample(n)

  #Add dfOutput to the global environment using the str as the name of the data frame
  assign(name, data.frame(dfOutput), envir = .GlobalEnv)

}
```

Using a for loop, create new data frames for each mouse with the help of the new function.

```
set.seed(6210)

for (name in SC_Names) {

  Randomly_Sample_Columns(dfData, name, 3)

}

#Combine these new data frames to use for the remainder of the analysis.
dfData <- cbind(dfSC1_, dfSC2_, dfSC4_, dfSC6_, dfSC5_, dfSC7_, dfSC9_, dfSC10_)
```

This order is dependent on the type of sample; the first four are tumor and the last four are luminal.

```
#Remove the data frames no longer needed.
rm(dfSC1_, dfSC2_, dfSC4_, dfSC5_, dfSC6_, dfSC7_, dfSC9_, dfSC10_,name)
```

Convert data frame into a DGE object for gene expression analysis.

This can be done using the DGEList function from edgeR. The input data frame is the one we just created, dfData, and the groupings are the cell type, so either luminal or tumor.

```
#This vector is to label each mouse sample type, luminal or tumor.
group <- rep(c("tumor", "luminal"), each = 12)

#Create the DGEList object.
DGE_Data <- DGEList(dfData, group = group)
#Check to see if this worked.
DGE_Data
```

```
## An object of class "DGEList"
## $counts
##        SC1_11 SC1_01 SC1_47 SC2_24 SC2_12 SC2_16 SC4_22 SC4_24 SC4_19 SC6_76
## Xkr4        0      0      0      0      0      0      0      0      0      0
## Rp1         0      0      0      0      0      0      0      0      0      0
## Sox17       0      0      0      0      0      0      0      0      0      0
## Mrpl15     36     13      0     26    190     11      3    131     27      2
## Lypla1      0     25      0      0     44     15      2      2     96      3
##        SC6_69 SC6_16 SC5_85 SC5_80 SC5_40 SC7_83 SC7_60 SC7_82 SC9_27 SC9_53
## Xkr4        0      3      0      0      0      0      0      0      0      0
## Rp1         0      3      0      0      0      0      0      0      0      0
## Sox17       0      0      0      0      0      0      0      0      0      0
## Mrpl15      0    480      7      0      1   1160    133      6    197      0
## Lypla1    631      0      4      1      0    163     31      9      0      0
##        SC9_68 SC10_13 SC10_37 SC10_09
## Xkr4        0       0       0       0
## Rp1         0       0       0       0
## Sox17       0       0       0       0
## Mrpl15      4     263       0       0
## Lypla1     21       1       2     510
## 20999 more rows ...
##
## $samples
```

```
##        group lib.size norm.factors
## SC1_11 tumor  1834673            1
## SC1_01 tumor  1945746            1
## SC1_47 tumor  1935799            1
## SC2_24 tumor  1872241            1
## SC2_12 tumor  2652039            1
## 19 more rows ...
```

```
#Can view each individual element of this list as well.
head(DGE_Data$counts, 5)
```

```
##        SC1_11 SC1_01 SC1_47 SC2_24 SC2_12 SC2_16 SC4_22 SC4_24 SC4_19 SC6_76
## Xkr4        0      0      0      0      0      0      0      0      0      0
## Rp1         0      0      0      0      0      0      0      0      0      0
## Sox17       0      0      0      0      0      0      0      0      0      0
## Mrpl15     36     13      0     26    190     11      3    131     27      2
## Lypla1      0     25      0      0     44     15      2      2     96      3
##        SC6_69 SC6_16 SC5_85 SC5_80 SC5_40 SC7_83 SC7_60 SC7_82 SC9_27 SC9_53
## Xkr4        0      3      0      0      0      0      0      0      0      0
## Rp1         0      3      0      0      0      0      0      0      0      0
## Sox17       0      0      0      0      0      0      0      0      0      0
## Mrpl15      0    480      7      0      1   1160    133      6    197      0
## Lypla1    631      0      4      1      0    163     31      9      0      0
##        SC9_68 SC10_13 SC10_37 SC10_09
## Xkr4        0       0       0       0
## Rp1         0       0       0       0
## Sox17       0       0       0       0
## Mrpl15      4     263       0       0
## Lypla1     21       1       2     510
```

```
head(DGE_Data$samples, 5)
```

```
##        group lib.size norm.factors
## SC1_11 tumor  1834673            1
## SC1_01 tumor  1945746            1
## SC1_47 tumor  1935799            1
## SC2_24 tumor  1872241            1
## SC2_12 tumor  2652039            1
```

```
#Extract column names for later analysis.
samplenames <- colnames(DGE_Data$counts)

#Remove variables that are no longer needed.
rm(dfData)
```

```
#Check to see if there any any duplicated gene names present in our data set.
anyDuplicated(rownames(DGE_Data$counts))
```

```
## [1] 0
```

As the data set I acquired already had gene names for each row, I did not need to extract these values using the Mus.musculus package.

There are also various gene names in the data set that consist of numbers and then the name Riken; for example, 4931408C20Rik. I do not have much experience working with the mouse genome but according to my research, these belong to names of genes that have not been officially annotated (Hayashizaki, 2003). Riken seems to be a large genomic institute in Japan that collaborates with various other institutes across the world to form comprehensive research on multiple different organisms. When I enter these names in UniProt, the associated gene name are the Rik ones we see here (https://www.uniprot.org/uniprot/E9PWP9).

In general for gene expression analysis, raw counts are not considered accurate values for comparison. Instead, counts per million (cpm) or log2-counts per million (lcpm) are utilized. This allows for the library size to be considered in the creation of these values. Library size is the total number of counts from each sample.

```
#The total library size for our subset data set is:
sum(DGE_Data$samples$lib.size)
```

```
## [1] 39624581
```

```
#The average library size for each sample is:
mean(DGE_Data$samples$lib.size)
```

```
## [1] 1651024
```

```
#Will display library size as a bar plot a little later on.
```

Converting the counts to cpm and lcpm.

```
dfCPM <- cpm(DGE_Data)
dfLCPM <- cpm(DGE_Data, log = T)
#Let's look at them.
head(dfCPM, 5)
```

```
##          SC1_11    SC1_01 SC1_47  SC2_24   SC2_12   SC2_16   SC4_22    SC4_24
## Xkr4    0.00000  0.000000      0  0.0000  0.00000 0.000000 0.000000  0.000000
## Rp1     0.00000  0.000000      0  0.0000  0.00000 0.000000 0.000000  0.000000
## Sox17   0.00000  0.000000      0  0.0000  0.00000 0.000000 0.000000  0.000000
## Mrpl15 19.62203  6.681242      0 13.8871 71.64299 5.326982 3.829119 95.525470
## Lypla1  0.00000 12.848542      0  0.0000 16.59101 7.264067 2.552746  1.458404
##          SC4_19   SC6_76   SC6_69     SC6_16   SC5_85    SC5_80    SC5_40
## Xkr4    0.00000 0.000000   0.0000   1.568508 0.000000 0.0000000 0.0000000
## Rp1     0.00000 0.000000   0.0000   1.568508 0.000000 0.0000000 0.0000000
## Sox17   0.00000 0.000000   0.0000   0.000000 0.000000 0.0000000 0.0000000
## Mrpl15 24.23557 3.100054   0.0000 250.961234 3.567230 0.0000000 0.9864774
## Lypla1 86.17091 4.650081 285.7334   0.000000 2.038417 0.6498515 0.0000000
##           SC7_83   SC7_60   SC7_82   SC9_27 SC9_53    SC9_68    SC10_13
## Xkr4     0.00000  0.00000 0.000000   0.0000      0  0.000000  0.0000000
## Rp1      0.00000  0.00000 0.000000   0.0000      0  0.000000  0.0000000
## Sox17    0.00000  0.00000 0.000000   0.0000      0  0.000000  0.0000000
## Mrpl15 598.89669 81.10092 3.596754 110.5619      0  4.288592 78.1582470
## Lypla1  84.15531 18.90322 5.395130   0.0000      0 22.515109  0.2971796
##         SC10_37  SC10_09
## Xkr4   0.000000   0.0000
## Rp1    0.000000   0.0000
## Sox17  0.000000   0.0000
## Mrpl15 0.000000   0.0000
## Lypla1 1.130883 962.7328
```

```
head(dfLCPM, 5)
```

```
##           SC1_11    SC1_01    SC1_47    SC2_24    SC2_12    SC2_16    SC4_22
## Xkr4   0.2766352 0.2766352 0.2766352 0.2766352 0.2766352 0.2766352 0.2766352
## Rp1    0.2766352 0.2766352 0.2766352 0.2766352 0.2766352 0.2766352 0.2766352
## Sox17  0.2766352 0.2766352 0.2766352 0.2766352 0.2766352 0.2766352 0.2766352
## Mrpl15 4.3808225 2.9804992 0.2766352 3.9163270 6.1869399 2.7089235 2.3335601
## Lypla1 0.2766352 3.8135121 0.2766352 0.2766352 4.1539945 3.0832841 1.9123074
##           SC4_24    SC4_19    SC6_76    SC6_69    SC6_16    SC5_85    SC5_80
## Xkr4   0.2766352 0.2766352 0.2766352 0.2766352 1.4750175 0.2766352 0.2766352
## Rp1    0.2766352 0.2766352 0.2766352 0.2766352 1.4750175 0.2766352 0.2766352
## Sox17  0.2766352 0.2766352 0.2766352 0.2766352 0.2766352 0.2766352 0.2766352
## Mrpl15 6.5959900 4.6694167 2.1081608 0.2766352 7.9782642 2.2565842 0.2766352
## Lypla1 1.4167138 6.4492654 2.5512543 8.1646259 0.2766352 1.7003414 0.8962457
##           SC5_40    SC7_83    SC7_60    SC7_82    SC9_27    SC9_53    SC9_68
## Xkr4   0.2766352 0.2766352 0.2766352 0.2766352 0.2766352 0.2766352 0.2766352
## Rp1    0.2766352 0.2766352 0.2766352 0.2766352 0.2766352 0.2766352 0.2766352
## Sox17  0.2766352 0.2766352 0.2766352 0.2766352 0.2766352 0.2766352 0.2766352
## Mrpl15 1.1360872 9.2290750 6.3630325 2.2654703 6.8044273 0.2766352 2.4594180
## Lypla1 0.2766352 6.4155977 4.3301671 2.7238826 0.2766352 0.2766352 4.5684226
##          SC10_13   SC10_37   SC10_09
## Xkr4   0.2766352 0.2766352 0.2766352
## Rp1    0.2766352 0.2766352 0.2766352
## Sox17  0.2766352 0.2766352 0.2766352
## Mrpl15 6.3105114 0.2766352 0.2766352
## Lypla1 0.5931580 1.2278932 9.9128023
```

You can see that in the cpm matrix, all the 0 gene expression values remained 0 but in the lcpm matrix, this has been transformed to a more relational data set; it is useful when creating exploratory plots. It reduces the inter-sample changes and prevents a large separation between the lowest and highest count values.

Calculating the L and M parameters used in the lcpm calculations; it will be used for generating read density figures later on.

```
L <- mean(DGE_Data$samples$lib.size) * 1e-6
M <- median(DGE_Data$samples$lib.size) * 1e-6
```

The library size for this data set is very very low in comparison to the vignette data set. This has about 1.6 million on average while the vignette data set was at 45.5 million. We will see how this is affected when we compare the genes lost during filtration in the next step.

Create a bar plot showing the difference in library size.

The average library size for each mouse. I was going to make this via lapply but I was confused as to how to change the ranges with each iteration without using a for loop.

```
SC1_lib <- mean(DGE_Data$samples$lib.size[1:3])
SC2_lib <- mean(DGE_Data$samples$lib.size[4:6])
SC4_lib <- mean(DGE_Data$samples$lib.size[7:9])
SC6_lib <- mean(DGE_Data$samples$lib.size[10:12])
SC5_lib <- mean(DGE_Data$samples$lib.size[13:15])
SC7_lib <- mean(DGE_Data$samples$lib.size[16:18])
```

```r
SC9_lib <- mean(DGE_Data$samples$lib.size[19:21])
SC10_lib <- mean(DGE_Data$samples$lib.size[22:24])

#Create a data frame containing the information needed to create a visualization.
dfLib <- data.frame(name = SC_Names,
                    size = rbind(SC1_lib, SC2_lib, SC4_lib, SC5_lib,
                                 SC6_lib, SC7_lib, SC9_lib, SC10_lib),
                    cell_type = c("tumor","tumor", "tumor", "luminal",
                                  "tumor", "luminal", "luminal", "luminal"))

dfLib %>% ggplot(aes(x = factor(name, levels = SC_Names),
                     y = size,
                     fill = cell_type)) +
  geom_bar(stat = "identity") +
  scale_fill_manual(name = "Cell-type",
                    values = c("#47E0E5", "#62E547")) +
  theme_bw() +
  labs(x = "Mouse Name",
       y = "Library Size",
       title = "Library Size in Samples")
```

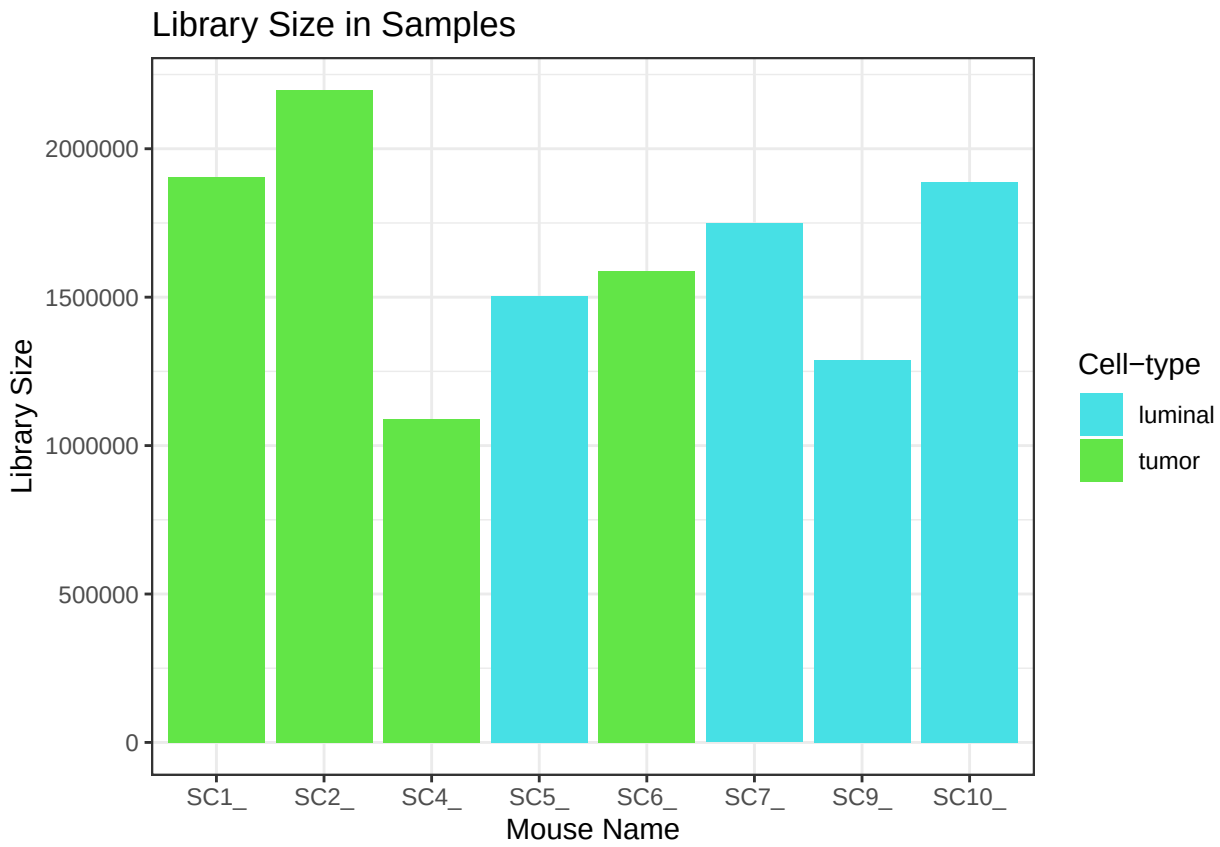


Figure 1: Bar plot displaying average library size per mouse.

The distribution of these library sizes on average per cell-type is not noticeably different. There is one sample from each that has a lower library size; this can be due to the fact that I randomly selected three samples

from each mouse. If we had used the full data set, we may not see this distribution.

I want to see the average library size for each grouping and see if we have similar numbers.

```
#Luminal
mean(SC5_lib, SC7_lib, SC9_lib, SC10_lib)
```

```
## [1] 1504943
```

```
#Tumor
mean(SC1_lib, SC2_lib, SC4_lib, SC6_lib)
```

```
## [1] 1905406
```

Their average is very similar so we won't worry about the individual variation.

Let's determine if there are any genes that have zero expression in all samples.

```
table(rowSums(DGE_Data$counts == 0) == 24)
```

```
##
## FALSE  TRUE
## 13575  7429
```

There are 7429 genes with zero expression. These will be deleted from our data set as they can hinder our downstream analysis.

The function, filterByExpr from the edgeR package, allows us to filter out the lowly expressed genes while still maintaining a large amount of data for analysis. It creates a logical vector displaying the rows to be kept and removed.

```
KeepExprs <- filterByExpr(DGE_Data, group = group)
DGE_Data <- DGE_Data[KeepExprs,, keep.lib.sizes = F]
```

Let's compare the amount of genes we lost to the number that had zero counts in all samples.

```
dim(DGE_Data)
```

```
## [1] 3382   24
```

```
(21004-3382)/21004 * 100
```

```
## [1] 83.89831
```

It has removed a *substantial* amount of genes! After re-analyzing the original data set, I came to realize that most of the rows had values of 0 throughout each gene. This could be because I have only taken a small subset of the samples actually analyzed by the authors of the paper or it could be the due to the quality of reads attained by the authors while carrying out the experiment. I am going to continue my analysis using the 3382 genes I do have in this data as it should still be able to yield functional plots for my interpretation. If I am obstructed with errors, I will have to re-evaluate this data set or the filtration step itself.

I will re-try this step and reduce the minimum count to 5 as the default is set to 10. I hope we notice a fewer amount of genes being discarded.

```
# KeepExprs <- filterByExpr(DGE_Data, group = group, min.count = 5)
# DGE_Data <- DGE_Data[KeepExprs,, keep.lib.sizes = F]
# dim(DGE_Data) #3914 24
# (21004-3914)/21004 * 100 #81.4%
```

As the number of samples barely increased, I will keep the default minimum count as it is recommended by the vignette for a more accurate downstream analysis. The vignette also states that the genes retained have counts in mostly all samples for the same grouping. In our case, these groupings would be luminal and tumor. So if a gene has multiple zero counts for all the luminal and tumor samples, this gene would be disregarded. If the gene is of interest to the study, most of the samples from the same groupings should have a count associated. If not, then the count for that gene could be a rogue value. This would mean that the expression of the particular gene cannot depict a reliable conclusion as it may not play a major role in our analysis. After reading this, I re-checked to make sure I made the correct groupings based on the paper and the GEO database; as the authors did not specifically specify what each sample corresponds to, my speculations could be inaccurate. Furthermore, the vignette data set lost more than 60% of their data during this filtration step so my assumptions could be the right choice. They also mention that a lower library size can be a factor of losing more data as there is less information to evaluate. Let's move on and see what our current data set can provide us.

Remove variables that are not required.

```
rm(SC1_lib, SC2_lib, SC4_lib, SC5_lib, SC6_lib, SC7_lib, SC9_lib, SC10_lib, KeepExprs)
```

Comparison of density reads - before and after filteration

I will now produce a figure comparing the density of reads from the raw unfiltered data and the filtered data. This code is adapted from the vignette for RNA-Seq analysis by Bioconductor.

```
lcpm.cutoff <- log2(10/M + 2/L)
nsamples <- ncol(DGE_Data)
col <- colorRampPalette(brewer.pal(12, "Paired")) (nsamples)
#Add more colors to the palette.
par(mfrow=c(1,2))
plot(density(dfLCPM[,1]),
     col=col[1],
     lwd=2,
     ylim=c(0,2),
     las=2, main="", xlab="")
title(main="A. Raw data", xlab="Log-cpm")
abline(v=lcpm.cutoff, lty=3)
for (i in 2:nsamples){
  den <- density(dfLCPM[,i])
  lines(den$x, den$y, col=col[i], lwd=2)
}
legend("topright", samplenames, text.col=col, bty="n", cex = 0.5)

dfLCPM_filtered <- cpm(DGE_Data, log=TRUE)
plot(density(dfLCPM_filtered[,1]),
     col=col[1],
     lwd=2,
     ylim=c(0,2),
     las=2,
```

```
     main="", xlab="")
title(main="B. Filtered data", xlab="Log-cpm")
abline(v=lcpm.cutoff, lty=3)
for (i in 2:nsamples){
  den <- density(dfLCPM_filtered[,i])
  lines(den$x, den$y, col=col[i], lwd=2)
}
legend("topright", samplenames, text.col=col, bty="n", cex = 0.5)
```

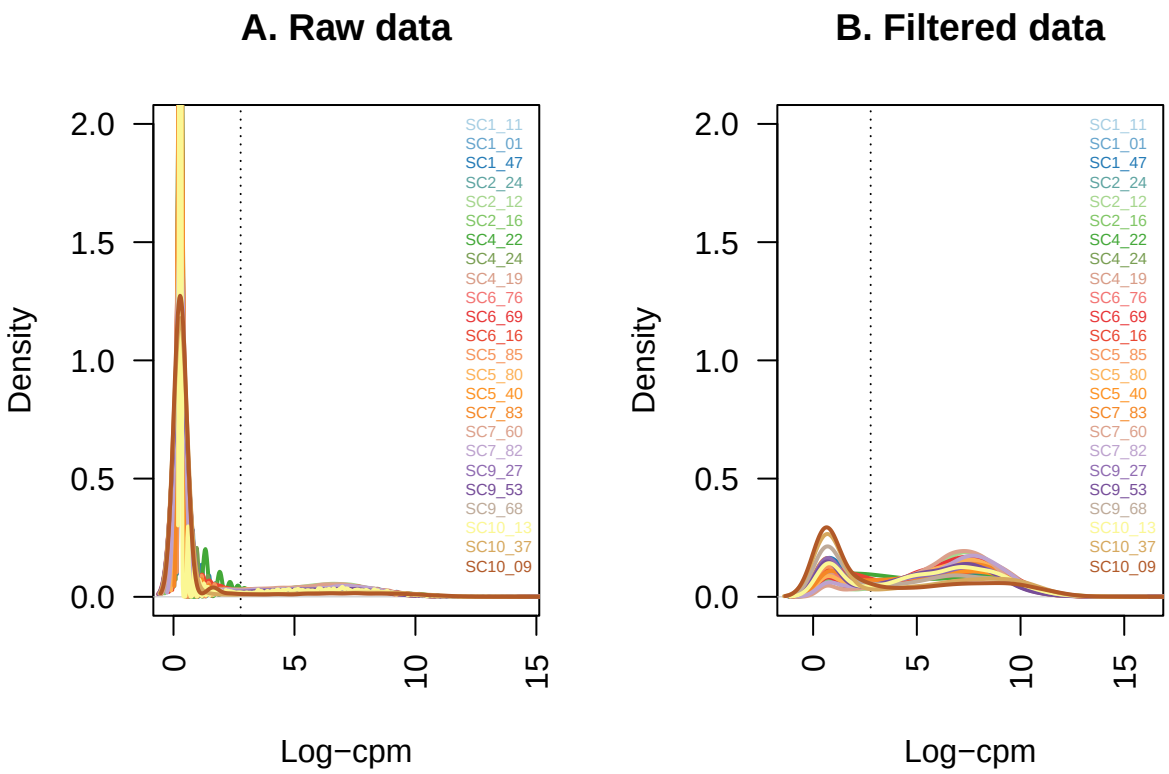


Figure 2: Read density comparison plot. A. Shows the unfiltered data; can see a large amount of reads at 0. B. Shows the filtered data; most of the lowly expressed genes have been removed.

We can see a drastic difference between the raw data and the filtered. A large amount of data were at zero to low values. We can also see a varying expression for each sample; this can account to the quality of the data from this study and the difference in their individual library size.

Normalization

Normalization is carried out to allow for a reasonably similar expression pattern among all the samples being analyzed. As we saw in the bar plot, there were a few samples with lower library size and one with almost half of the largest library. This can occur from external factors during the experiment such as sequencing depth and even GC count (Evans *et al.*, 2017). The main reason for normalization is to showcase the true differences in samples instead of using the raw values given by the instrument (Evans *et al.*, 2017). Another reason for the use of normalization is that the variance in gene counts can be from the outcome of differential coverage instead of differential gene expression (Evans *et al.*, 2017). The vignette uses the method of Trimmed Mean of the M-values (TMM) where a sample is chosen as a reference and fold-changes are calculated relative to this sample (Evans *et al.*, 2017). Apparently this method of usage is known only from being used in the

limma package, which is the one we are using today. This method did not give a good representation of normalized data so I chose TMMwsp instead which is 'TMM with singleton pairing' method. This performs better with data consisting of multiple 0 expression values; as we have seen, this data set is riddled with 0 expression, hence it works better for my further analysis.

```
DGE_Data <- calcNormFactors(DGE_Data, method = "TMMwsp")
#We see a new element in our DGE_Data object called norm.factors.
DGE_Data$samples$norm.factors
```

```
##  [1] 1.2589184 1.1132583 0.7839350 0.9963920 1.3155666 1.1674357 0.8376471
##  [8] 0.9347514 1.5211617 0.9395069 1.1306151 1.1053613 0.9047118 1.0784354
## [15] 0.7651143 1.0233774 1.2814494 1.4029791 0.6791346 0.7757668 0.7609038
## [22] 0.9136804 0.8834961 0.9568938
```

To better visualize the impact of normalization, we will create a boxplot showcasing log-CPM values of expression distribution for both unnormalized and normalized data. We will see that in the unnormalized data, there will be a variance in expression levels but this will be standardized in the normalized image. The DGE object is duplicated to manipulate the already normalized data into showcasing what unnormalized data looks like. I have actually commented the plots as I believe other plots in this section better showcase what I would like to portray for this assignment.

```
# DGE_Data_2 <- DGE_Data
# DGE_Data_2$samples$norm.factors <- 1 #Setting all the norm factors to 1
# DGE_Data_2$counts[,1] <- ceiling(DGE_Data_2$counts[,1] * 0.05)
# DGE_Data_2$counts[,2] <- DGE_Data_2$counts[,2] * 5
#
# par(mfrow = c(1,2))
# dfLCPM <- cpm(DGE_Data_2, log = T)
# boxplot(dfLCPM, las = 2, col = col,
#         main = "A. Unnormalizaed Data Example",
#         ylab = "Log-cpm") #Using the same colors as the plot above
# DGE_Data_2 <- calcNormFactors(DGE_Data_2, method = "TMMwsp")
# Normalizing data in this duplicated DGE object
# dfLCPM <- cpm(DGE_Data_2, log = T)
# boxplot(dfLCPM, las = 2, col = col,
#         main = "A. Normalizaed Data Example",
#         ylab = "Log-cpm")
# par(mfrow = c(1,1))
```

When plotted, we can see that the mean values of all the samples still do not line up as well as the example data set in this vignette, however, this could be a result of the quality of data collection done by the authors of the paper or even the mice themselves. We can see a noticeable difference in mouse 10 compared to the others. As the authors did not lay out the exact differences between the mice tested (gestation period), my speculation is that this mouse was affected by some other external factor compared to the other mice in its category.

I had also created a multi-dimensional scaling(MDS) visualization to display the similarities and differences in the samples being used. This was done using the plotMDS function in limma. However, I decided to delete the plot from the assignment as there was no clear distinction between the two groupings of the samples. Two of the tumor samples were grouped together with the luminal samples; as I mentioned before, I am unable to differentiate the gestation period of each mouse sample, so this could be a reason as to why two samples grouped irregularly. This distinction is also seen in Figure 4 where two of the tumor samples have clustered with the luminal samples.

```
rm(dfLCPM_filtered, dfLib, L, M, SC_Names, samplenames)
```

Main Software Tools

The main vignettes I decided to adapt for this script are both from Bioconductor: RNA-seq analysis is easy as 1-2-3 with limma, Glimma and edgeR, and goseq: Gene Ontology testing for RNA-seq data sets.

- https://master.bioconductor.org/packages/release/workflows/vignettes/RNAseq123/inst/doc/limmaWorkflow.html
- https://bioconductor.org/packages/devel/bioc/vignettes/goseq/inst/doc/goseq.pdf

To begin, I researched different RNA sequencing packages available and used the paper by Seyednasrollah *et al.* as a guideline for selecting a specific software tool. To explain briefly, the paper outlines the most popular software tools currently used for analyzing RNA-sequencing data. The authors used real data to conduct their analysis instead of using simulated data; this accounts for natural variability seen in real-life data and allows the comparison to be more realistic. As the human understanding of variability is imperfect, using this type of real data becomes a large advantage while comparing computational tools. Through their analysis, I decided to use the tool limma by Bioconductor as this package had a high precision level which corresponds to the accuracy of analyzing replicates. The authors also display a proportion of false discoveries for the software tools analyzed and limma displays a low proportion. As I also considered using edgeR by Bioconductor, I decided not to choose this as it shows a varying degree of false discovery proportion between sample types as well as lower precision levels when analyzing replicates. Both limma and edgeR are available packages by Bioconductor and I utilize both of these when conducting my analysis. Sun *et al.* use Cufflinks to carry out their RNA-seq analysis and TopHat for alignment. My main analysis is using the function voom by limma which uses linear regression modeling while analyzing the read counts for the RNA sequence data (Law et. al., 2014). The functions I use from edgeR are DGEList which helps me covert my data frame downloaded from GEO into a DGEList object and filterByExpression which allows the data to filter out the lowly expressed genes. I also use the package geoseq by Bioconductor to carry out my Gene Ontology analysis which accounts for the gene length bias of our differentially expressed genes. Understanding this vignette and carrying out the analysis was a very new experience for me. It was thoroughly an educational experience and helped me understand the difficulties of using new data objects. I consulted many vignettes and tutorials to be able to carry out the analysis seen in Figure 5 of this script. However, this journey has encouraged me to develop new investigative skills and has me excited for my future summer project!

Main Analysis

The main analysis of this script is showcasing the genes that are differentially expressed among the two groupings of our data, tumor and luminal. For this, we are in the assumption that our data is normally distributed. First we create a design matrix for the cell-type groupings. I used the vignette (https://bioconductor.org/packages/release/workflows/vignettes/RNAseq123/inst/doc/designmatrices.html) to help me determine if I need to use an intercept term. I will not include an intercept term as the grouping are not covariates and the models are considered equivalent.

```
mtDesign <- model.matrix(~0+group)
colnames(mtDesign) <- gsub("group", "", colnames(mtDesign))
head(mtDesign)
```

```
##   luminal tumor
```

```
## 1       0     1
## 2       0     1
## 3       0     1
## 4       0     1
## 5       0     1
## 6       0     1
```

We have to set up model contrasts for the our groups as they do not have an associated intercept. The makeContrasts function from limma is used to make pairwise comparison between the two cell populations.

```
mtContrasts <- makeContrasts(TvsL = tumor - luminal, levels = colnames(mtDesign))
mtContrasts
```

```
##          Contrasts
## Levels    TvsL
##   luminal   -1
##   tumor      1
```

Honestly, I found going along the vignette quite difficult for my data set. Each step took a few hours to determine if my arguments are entered correctly or if I have to manipulate my data to input the correct class or even if my results make sense. As I am not completely sure if my groupings are accurate, I am always worried that my results are incorrect. However, as this assignment is about conducting an analysis along with learning the ups and downs about data sets and packages, I believe that I am learning a lot along the way.

Removing heteroscedasticity from count data

When using raw counts for RNA-seq data, the variance is not considered independent from the mean. For this script, we use the lcpm values where we assume our data is normally distributed. The function, voom, calculates precision weights on the mean-variance dependency by using library size and the normalization factors. I have commented out the plots as I believe the other plots I have created in the main analysis portray more significant visualizations for this assignment.

```
#par(mfrow = c(1,2))
V_EList <- voom(DGE_Data, mtDesign, plot = F, save.plot = T)
#plot(V_EList$voom.xy,
#     xlab = "Average log of counts",
#     ylab = "Sqrt (standard deviation)",
#     main = "A. voom: Mean-variance trend",
#     pch = 20, cex = 0.1)
#lines(V_EList$voom.line, col = "red")
VFit <- lmFit(V_EList, mtDesign)
VFit <- contrasts.fit(VFit, contrasts = mtContrasts)
EFit <- eBayes(VFit)
#plotSA(EFit, main = "B. Final Model: Mean-variance trend",
#       ylab = "Sqrt (sigma)",
#       xlab = "Average log of expression")
```

In these plots, the means are plotted on the x axis and the variances are plotted on the y axis. After briefly reading the paper introducing voom (Law *et al.*, 2014), I was unable to understand why the voom plot has a slight increase trend before decreasing. My understanding of linear models is not adequate enough to have an explanation; if this can be explained, I would really like to understand why. In my research, I have

come across the downwards trend correlating to inaccurate groupings or the need to filter the data higher. The black dots in each of these plots correlates to a gene in our data set. We can see that this data set is smaller in comparison to the data set used in the vignette. Voom creates an EList object containing various information we have already seen in the DGEList object as well as additional information such as expression values and precision weights.

Examining the number of differentially expressed genes

A simple table can be created using the decideTests function from limma to summarize the number of significantly up- and down-regulated genes. As per usual, significance is cut off at a p-value of 0.05 by default.

```
dt <- decideTests(EFit)
summary(dt)
```

```
##        TvsL
## Down     26
## NotSig 3328
## Up       28
```

We can see that most of the genes are not significantly different in the two groupings. This tells us that there are 28 genes that are up-regulated in the tumor samples and 26 down-regulated genes in the tumor samples relative to the luminal samples. This can mean that there only a few gene expressions that are affected in our data set when comparing tumor and luminal mammary samples of mice.

This step allows us to extract the differentially expressed gene names. As 0 represents genes that are not differentially expressed, we are looking for the rest.

```
DE_Common <- which(dt[,1] != 0)
length(DE_Common) #54 genes
```

```
## [1] 54
```

```
DE_Genes <- rownames(EFit$coefficients)[DE_Common]
DE_Genes
```

```
##  [1] "Dbi"       "Cytip"     "Muc15"     "Spred1"    "Ndrg3"     "Serinc3"
##  [7] "Pex2"      "Tm4sf1"    "Txnip"     "Cttnbp2nl" "Hsdl2"     "Nfib"
## [13] "Nudc"      "Ppp1cb"    "Csn1s1"    "Csn2"      "Cxcl1"     "Calu"
## [19] "Creb5"     "Mgst1"     "Stk38l"    "Tbcb"      "Emsy"      "Ifitm3"
## [25] "Arhgef7"   "Trmt1"     "Pkm"       "Gja1"      "Cd63"      "Wfdc18"
## [31] "Stat5a"    "Syngr2"    "Lpin1"     "Slc28a3"   "Dapk1"     "Anxa7"
## [37] "Gjb2"      "Selenop"   "Ext1"      "Fam91a1"   "Mtss1"     "Puf60"
## [43] "Med15"     "Cd200"     "Sod2"      "H2-K1"     "H2-Q6"     "Ubd"
## [49] "Clpp"      "C3"        "Dsg2"      "Snx2"      "Ltbp3"     "Scd1"
```

These are the significantly expressed genes. I am storing this vector for later use for finding GO terms.

Displaying the differential expression results graphically

Limma has a function called plotMD which allows for the genes to be displayed visually by mean-difference plots. This uses log-FC values from our lineal model analysis against the mean lcpm value. Glimma creates this same plot using glMDPlot, however it creates an interactive html link which is better for deeper analysis!

```
dfLCPM <- cpm(DGE_Data, log = T)
glMDPlot(EFit, coef = 1,
         status = dt,
         main = "Tumor vs Luminal",
         side.main = "Genes",
         counts = dfLCPM,
         groups = group,
         launch = T)
```

HTML link to the interactive plot.

Make sure to save the glimma-plots folder in your working directory for this link to be active. If this does not work for you, the plot will appear in your web browser when the script is run. Here's a screenshot for a preview.

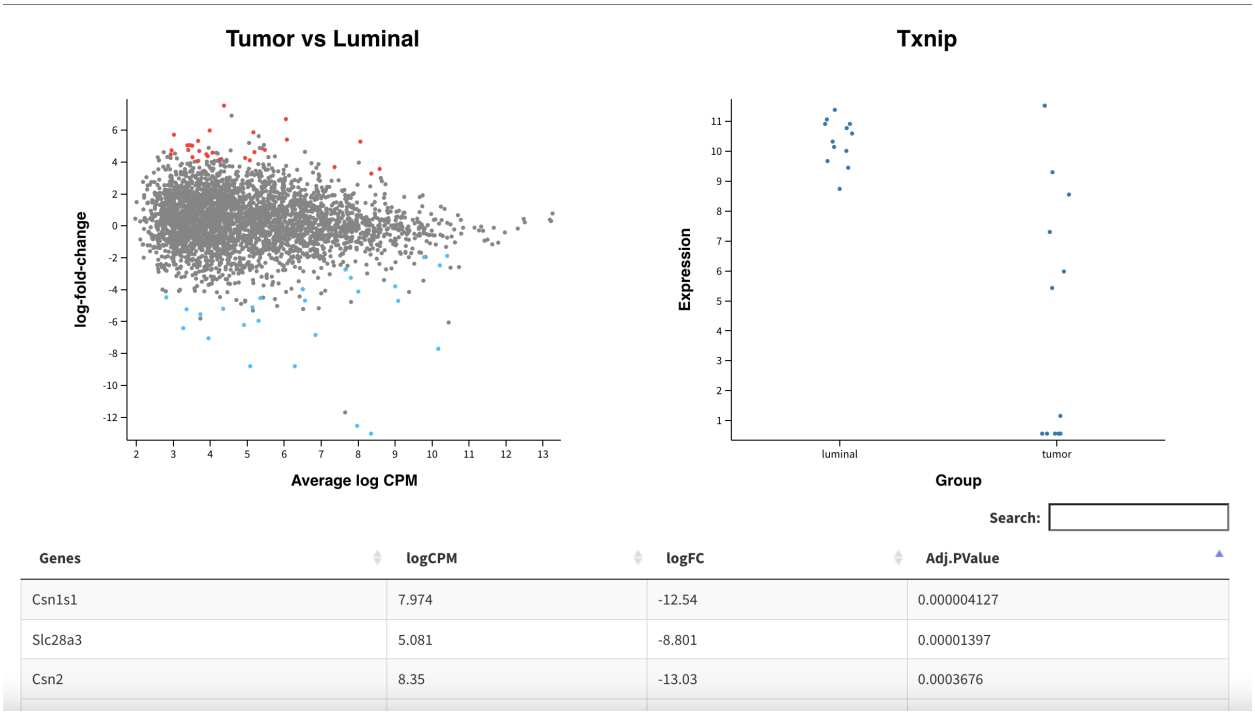


Figure 3: A screenshot preview of the interactive MD plot.

I also created this same plot using limma's plotMD function. It has been commented out as the interactive plot by glimma is more visually appealing.

```
par(mfrow = c(1,1))
# plotMD(EFit, column = 1, status = dt[,1],
#        main = "Tumor vs Luminal",
```

```
#         xlim = c(0, 13),
#         xlab = "Average log of expression",
#         ylab = "logFC")
```

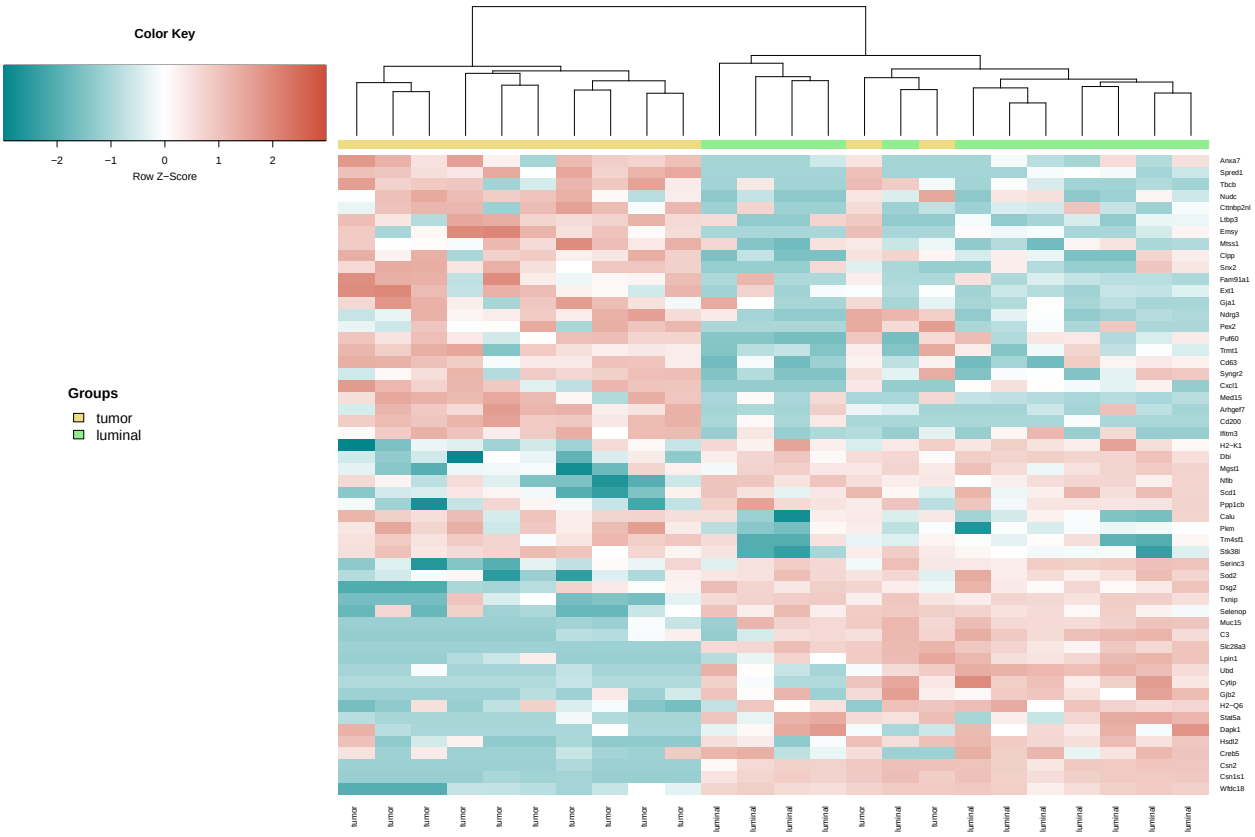
A MD plot is an interpretation of the Bland-Altman plot which calculates the agreement between quantitative measures (Giavarina, 2015). The mean and sd of the differences between the measurements are used to calculate the statistical limits (Giavarina, 2015). It is a scatter plot where the x axis displays average of the means and the y axis measures the difference of the paired measurements we calculated earlier when creating our contrast matrix (Giavarina, 2015). As we can see in the colored dots, the red ones display the up-regulated genes and the blue dots show the down-regulated genes.

A heatmap can also be used to display the differentially expressed genes. We will be using all 54 DE genes. We can utilize the heatmap.2 function from the gplots package to create this.

```
dfLCPM <- cpm(DGE_Data, log = T)
Color_HeatMap <- colorpanel(1000, "turquoise4", "white", "tomato3")

heatmap.2(dfLCPM[DE_Common,],
          scale = "row",
          labRow = DE_Genes,
          labCol = group,
          key = T,
          key.par = list(mar = c(0,1,5,1)),
          col = Color_HeatMap,
          trace = "none",
          density.info = "none",
          lhei = c(2,10),
          dendrogram = "column",
          ColSideColors = rep(c("lightgoldenrod2", "palegreen2"),
                              each = 12))

legend("left",
       title = as.expression(bquote(bold("Groups"))),
       legend = c("tumor", "luminal"),
       fill = c("lightgoldenrod2", "palegreen2"),
       cex = 1,
       box.lty = 0,
       title.adj = 0.2)
```


Color Key
−2
−1
0
1
2
Row Z−Score
Groups
tumor
luminal
Anxa7
Spred1
Tbcb
Nudc
Ctnnbp2nl
Ltbp3
Emsy
Mtss1
Clpp
Snx2
Fam91a1
Ext1
Gja1
Ndrg3
Pex2
Puf60
Trmt1
Cd63
Syngr2
Cxcl1
Med15
Arhgef7
Cd200
Ifitm3
H2−K1
Dbi
Mgst1
Nfib
Scd1
Ppp1cb
Calu
Pkm
Tm4sf1
Slk3l
Serinc3
Sod2
Dsg2
Txnip
Selenop
Muc15
C3
Slc28a3
Lpin1
Ubd
Cyfip
Gjb2
H2−Q6
Stat5a
Dapk1
Hsdl2
Creb5
Csn2
Csn1s1
Wfdc18
tumor
tumor
tumor
tumor
tumor
tumor
tumor
tumor
tumor
tumor
luminal
luminal
luminal
luminal
tumor
luminal
tumor
luminal
luminal
luminal
luminal
luminal
luminal
luminal

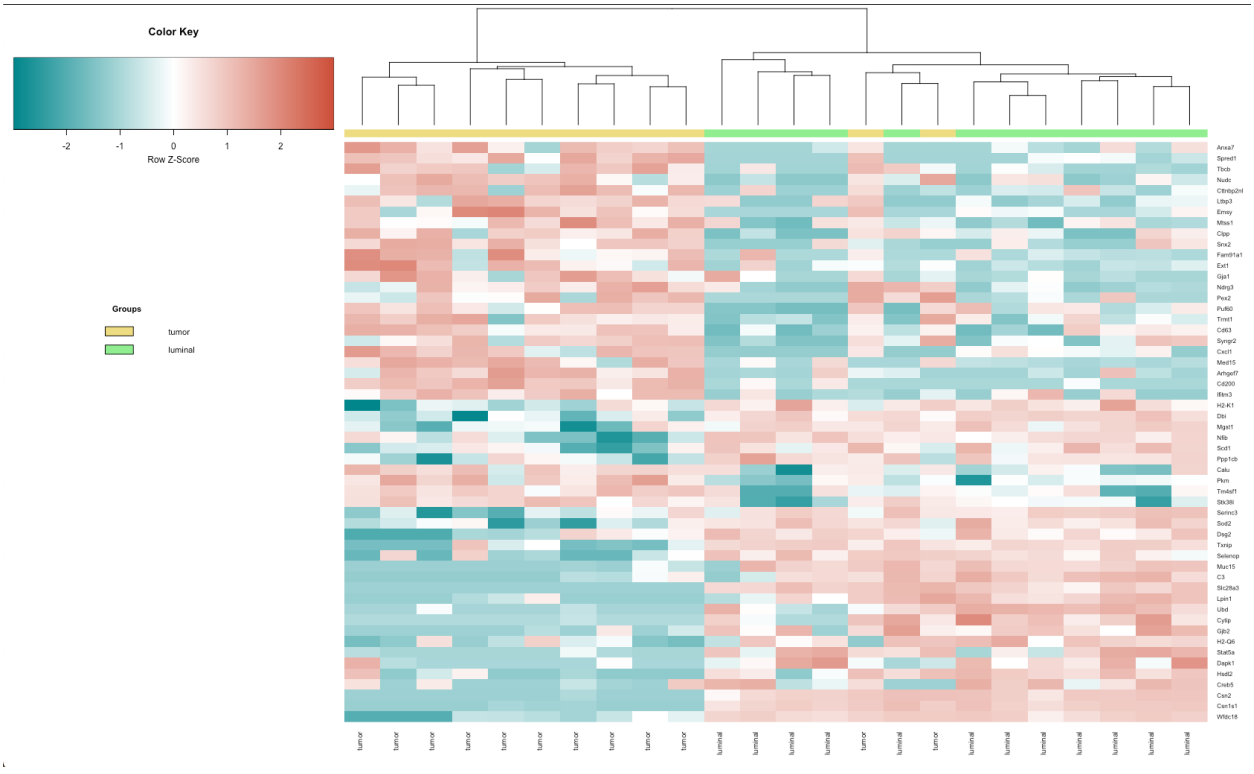


Figure 4: Heat map showing the differentially expressed genes. The first image is incomplete in RMarkdown. The second image is a screenshot from the plot created from my .R file.

Side note, it took me over two hours to get this right..why are legends so difficult? Also, I googled for quite a while trying to see why the image is incomplete on RMarkdown, but to no prevail. Do you have any advice?

As we can see, this data set is quite a mess. There are two tumor samples that have clustered with the luminal samples. As I talked about in the data acquisition section, both these cell-types are from a BRCA1-deficient mouse; the paper was studying the intratumor variations so a very clear distinction of gene expression is not going to be seen. However, we can mildly see a distinction between the up-regulated and down-regulated genes in the tumor and luminal samples. Most of the genes at the top of the plot show an up-regulation in tumor cells and the most of the genes at the bottom of the plot show an up-regulation in the luminal cells. The next step for my analysis is to correlate these differentially expressed genes with their associated GO terms.

Gene Ontology Enrichment Analysis

To perform this analysis, I have to create a named vector containing all the genes analyzed in this study and if there are differentially expressed. I will start by creating a vector containing the names of the 3382 genes we used.

```
Assayed_Genes <- rownames(dfLCPM)
```

DE_Genes is a vector we already created to showcase the differentially expressed genes. It consists of 54 genes.

After looking through the supportedOrganisms() list provided by goseq, I realized that the newest genome for Mus musculus (mm39) is not available through the offline database of goseq, hence I

needed to install the Bioconductor package for this specific genome and all its gene annotations (TxDb.Mmusculus.UCSC.mm39.refGene). I also had to go through the vignette for Genomic Feautures (https://bioconductor.org/packages/release/bioc/vignettes/GenomicFeatures/inst/doc/GenomicFeatures.pdf) as it helped with extracting the data I required from the TxDb object.

As the information I have is the gene names (SYMBOL), I have to use the Mus.musculus package to incorporate the gene names into the TxDb object. I am switching the mm10 TxDb from the Mus.musculus package to the newer mm39 TxDb.

```
TxDb(Mus.musculus) <- TxDb.Mmusculus.UCSC.mm39.refGene
Mus.musculus
```

```
## OrganismDb Object:
## # Includes GODb Object:  GO.db
## # With data about:  Gene Ontology
## # Includes OrgDb Object:  org.Mm.eg.db
## # Gene data about:  Mus musculus
## # Taxonomy Id:  10090
## # Includes TxDb Object:  TxDb.Mmusculus.UCSC.mm39.refGene
## # Transcriptome data about:  Mus musculus
## # Based on genome:  mm39
## # The OrgDb gene id ENTREZID is mapped to the TxDb gene id GENEID .
```

We can see that the TxDb object has been replaced from mm10 to mm39.

Obtaining the Gene ID's for each gene vector.

```
dbGeneID <- AnnotationDbi::select(Mus.musculus,
                                  keys = Assayed_Genes,
                                  columns = "GENEID",
                                  keytype = "SYMBOL")
dbDE_GENEID <- AnnotationDbi::select(Mus.musculus,
                                     keys = DE_Genes,
                                     columns = "GENEID",
                                     keytype = "SYMBOL")
```

Create vectors with the GENEID for both the full list of genes and DE genes.

```
Assayed_Genes_ID <- as.vector(dbGeneID$GENEID)
DE_Genes_ID <- as.vector(dbDE_GENEID$GENEID)
```

Now we will incorporate these two vectors to output a numerical vector stating if the original list of genes were differentially expressed (1) or not differentially expressed (0).

```
Gene_Vector <- as.integer(Assayed_Genes_ID %in% DE_Genes_ID)
names(Gene_Vector) <- Assayed_Genes_ID

#Let's look to see if we did what we intended.
head(Gene_Vector, 20)
```

```
##  21399 108664  12421 319263  70675  73824  26754 211660 211673  72265  19989
##      0      0      0      0      0      0      0      0      0      0      0
##  66799  67923  70155  68002  68421 213109  19243  19076  98403
##      0      0      0      0      0      0      0      0      0
```

To carry out the rest of our GO analysis, we have to manually extract the gene lengths for each of these genes before fitting the probability weighting function (PWF).

```
dfTxDb_Gene_Length <- transcriptLengths(TxDb.Mmusculus.UCSC.mm39.refGene)
head(dfTxDb_Gene_Length, 10)
```

```
##    tx_id      tx_name gene_id nexon tx_len
## 1      1 NM_001355712   18777     8   2401
## 2      2    NM_008866   18777     9   2503
## 3      3 NM_001159750   21399    10   2668
## 4      4    NM_011541   21399    10   2671
## 5      5 NM_001159751   21399    10   2564
## 6      6 NM_001310442  108664    13   2009
## 7      7    NM_133826  108664    14   2063
## 8      8 NM_001204371   18387     4   4707
## 9      9 NM_001318735   18387     4   4707
## 10    10    NM_011011   18387     4   4677
```

```
#Extract the gene lengths for each of the genes we require.
dfTxDb_Gene_Length_Subset <- dfTxDb_Gene_Length %>%
  filter(gene_id %in% Assayed_Genes_ID)

#We can see that many genes have multiple entries.
#We will have to average the gene lengths for the repeated entries.
dfTxDb_Gene_Length_Subset %<>%
  dplyr::select(gene_id, tx_len) %>%
  group_by(gene_id) %>%
  summarise(mean_length = ceiling(mean(tx_len)))

head(dfTxDb_Gene_Length_Subset, 10)
```

```
## # A tibble: 10 x 2
##    gene_id   mean_length
##    <chr>           <dbl>
##  1 100019          17959
##  2 100034361        1887
##  3 100034726         521
##  4 100037258        5190
##  5 100040298         778
##  6 100040970         694
##  7 100169           4529
##  8 100169864        1106
##  9 100434           2448
## 10 100494           3101
```

```
dim(dfTxDb_Gene_Length_Subset)
```

```
## [1] 3331    2
```

We can see that not every gene in our Assayed_Genes have an entry in this data frame. As those genes cannot be analyzed for the rest of out script, I will delete those from Gene_Vector.

```
Assayed_Genes_ID <- as.vector(dfTxDb_Gene_Length_Subset$gene_id)
Gene_Vector <- as.integer(Assayed_Genes_ID %in% DE_Genes_ID)
names(Gene_Vector) <- Assayed_Genes_ID

pwf <- nullp(Gene_Vector, bias.data = dfTxDb_Gene_Length_Subset$mean_length, plot.fit = F)
summary(pwf$pwf)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.00100 0.01740 0.01782 0.01733 0.01782 0.01782
```

All the PWF values are lower than 0.018. PWF shows the probability of each gene being differentially expressed based on their length alone. My understanding from the plot created and the summary of values is that due to the small size of the data set and the very few DE genes, the calculation of PWF here cannot give us a lot of accurate information. When plotted, the points are so far apart and do not show any comprehensive trend for gene length. The plot somewhat shows when the length of the genes are longer, the genes are less likely to be differentially expressed (the line curves downwards, opposite from the vignette).

Let's conduct the GO enrichment analysis and see what happens.

I searched and tried for many hours to use use the mm39 mouse genome for this next step. I was unable to find a way to incorporate the new genome into this function. I used mm9 which is the latest mouse reference genome available on the goseq offline database. Hopefully over the next week or the new months (before my project), I can figure out how to utilize the newest reference genome. Here is the code I tried.

```
#GO_Results_1 <- goseq(pwf = pwf, genome = "mm39", id = "GENEID", test.cats = "GO:BP")
#Error message: Couldn't grab GO categories automatically.  Please manually specify.
```

As mm39 is not in the offline database of goseq, I would have needed to find the GO categories for each gene manually and create a vector to input into this function. Due to time restraints for this assignment, I decided to use the most recent genome available in the offline goseq database, mm9 to finish my analysis.

```
Gene_Vector2 <- as.integer(Assayed_Genes %in% DE_Genes)
names(Gene_Vector2) <- Assayed_Genes

pwf2 <- nullp(Gene_Vector2, "mm9", "geneSymbol", plot.fit = F)

GO_Results <- goseq(pwf2, "mm9", "geneSymbol", test.cats = "GO:BP")

#Plot the top 10 GO term hits.
GO_Results %>%
  top_n(10, wt = -over_represented_pvalue) %>% #Young et al., 2010.
  mutate(hitsPerc = numDEInCat*100/numInCat) %>%
  ggplot(aes(x = hitsPerc,
             y = term,
             color = over_represented_pvalue,
             size = numDEInCat)) +
  geom_point() +
  scale_x_continuous(breaks = c(0,10,20,30)) +
  expand_limits(x = 0) +
  labs(x ="Hits (%)",
       y = "GO Terms",
       color = "p-value",
       size = "Count",
```

```
      title = "GO Enrichment Analysis")+
theme(aspect.ratio = 1.7/1,
      axis.text.y = element_text(size = 8)) +
scale_y_discrete(labels = function(x) str_wrap(x, width = 20))
```

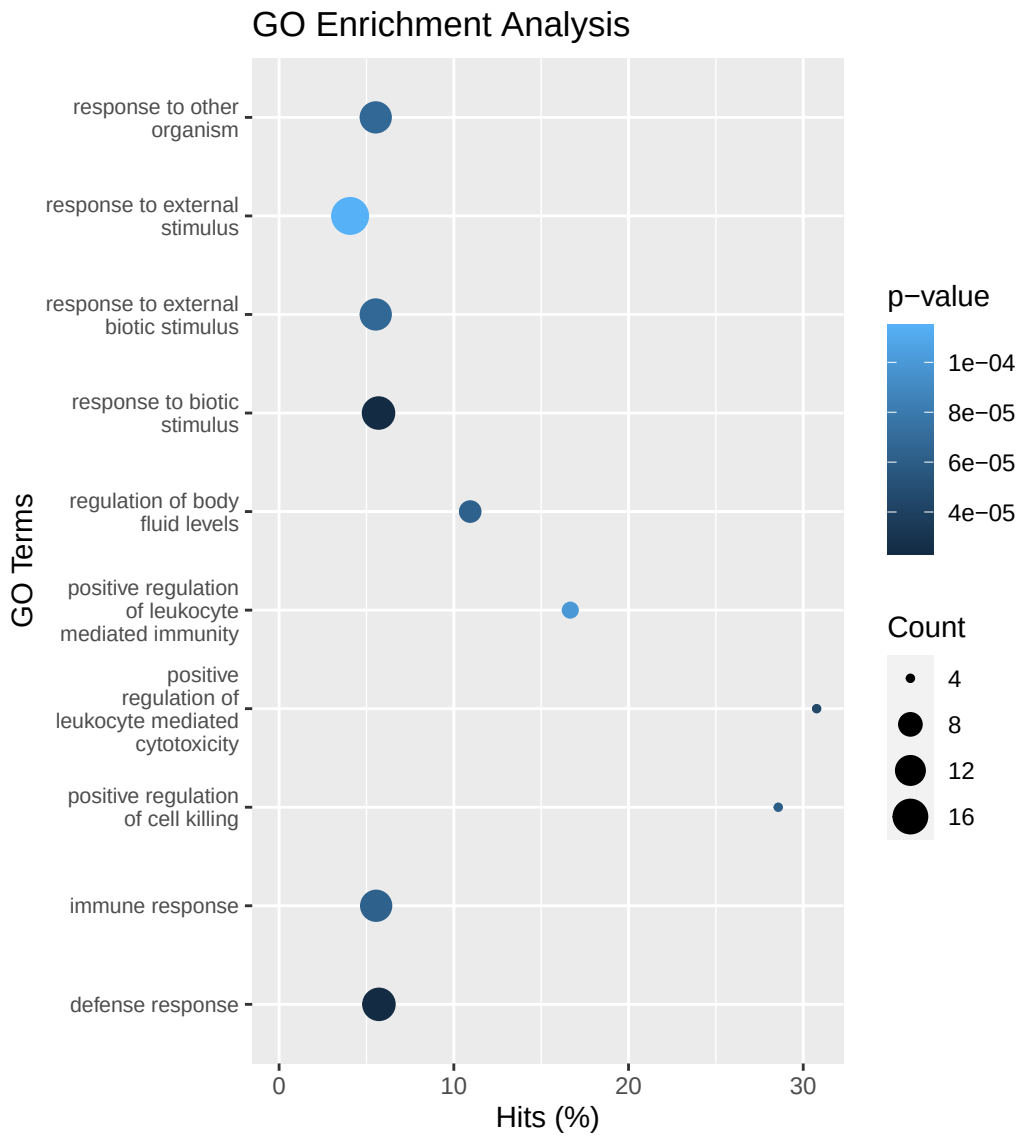


Figure 5: Gene ontology enrichment analysis. This plot shows the top 10 hits.

We can see that the top 10 GO terms seen in this image showcase a high magnitude immune response. I believe we can interpret this as the differentially expressed genes seen in our data set, showcase an effect seen on the immune response. This makes sense as we are analyzing the initiation and progression of tumor formation in the mice mammary glands. Functional profiling of enrichment analysis in this figure displays the biological process association seen in the differentially expressed genes. According to the official Gene Ontology website, p-value in enrichment analysis calculates the probability that the GO term is seen in the total list of genes inputted. The smaller the p-value, the closer the GO term is associated with the total genes. In our case, the darker the blue, the more significant is the term association in comparison to the whole list of GO terms.

Results and Discussion

Breast cancers that are formed due to a mutation in the BRCA1 gene, are mostly likely to have originated in the luminal epithelia of mammary glands (Sun *et al.*, 2021). This is the main reason the authors of the paper generated a RNA-seq workflow to study the differential gene expression between luminal and tumor mammary cells. This was achieved by using BRCA1-deficient mice and obtaining the required samples. The data set I used came from eight different mice, where four mice provided luminal samples and the other mice were used for tumor samples. In my opinion, a better approach would have been to obtain both luminal and tumor samples from each mouse for a more accurate comparison of differential expression. However, with the results we did obtain, we notice a small amount of genes that are seen to be significant. The question for this project was to analyze the differentially expressed genes and to determine if their functional roles were related to BRCA1 and its biological processes. From our heat map and Gene Ontology enrichment analysis, we can strongly suggest that the DE genes between the tumor and luminal samples do indeed have functional roles in cell repair and the immune response. I believe this portrays that the DE genes are related to the specific loss of BRCA1 and its biological functions. I used the list of differentially expressed genes in Figure 4 and inputted their GO ID's into the Gene Ontology online database to compare the results I received using goseq. As I had to use an older Mus musculus reference genome while programming with the goseq function, there were a few genes that were not found and hence not analyzed. The online analysis gave me the same result as Figure 5 with a few additional biological processes. These were, "animal organ development" and "system development"; these processes correspond to others related to "response to external stimulus" and "immune response"! Another process, with a slightly higher p-value, was "negative regulation of epithelial cell proliferation involved in lung morphogenesis". This tells us that most of our statistically significant biological matches correlate to the functions of BRCA1 and showcase how detrimental the loss of proper BRCA1 function can be to the organism. The DE genes affecting epithelial cell proliferation is also a contributing factor to the author's study; it shows that the luminal cells are indeed involved in the initiation of tumorigenesis. Some of the biological processes determined to be significant in Figure 5 were also found in Figure 3E by Sun *et al.* However, their gene enrichment analysis displayed a larger variety of biological processes affected by the loss of BRCA1 in the two cell types. According to the 2016 paper by Hein *et al.*, luminal cells respond to tumorigenesis by adapting to their surroundings and are involved in cellular heterogeneity within the tumor. This is an important area of research as most breast cancer patients develop tumors with a vast amount of luminal progenitor cells which can imply major involvement of luminal cells during the formation of mammary tumors (Hein *et al.*, 2016).

Even though the results of this analysis was successful in answering my original question, we still have to consider the caveats and drawbacks of the data itself and the methodology used. For the scope of this assignment, I had decided to use only three samples from each mouse tested so the sample size for each grouping was quite small in comparison to the analysis undertaken by Sun *et al.* Due to this, we can see a notable difference in the library size of each mouse shown in Figure 1. However, the average library size for each grouping was similar so I decided to continue my analysis on the subset data. Figure 2 shows the density read comparison before and after data filtration; we can see a large amount of lowly expressed genes in our data set. About 36% of the full data set consisted of zero expression of genes in all samples while the other 48% consisted of genes with low expression. As we calculated, this was a total of 83.9% of genes lost from our original data set and we were left with 3382 genes to analyze for the remainder of the script. As the vignette data set lost more than 60% of their data as well during this filtration step, I was not too concerned about the loss of data; however, their library size was exponentially larger than the one I used. This can also mean that the loss of data could be the result of something fundamental as poor sampling or minimal mice specimens. As each mouse was responsible for over 30 samples each, some of the samples themselves could be of insufficient quality. Another factor to consider is inaccurate grouping of the samples as well; as I only categorized the samples by luminal and tumor (which was the only distinction I could find), I was unable to take into account the gestation of the mice which would have been important when differentiating the groupings during gene expression analysis (voom). As I only had two groupings to work with, my analysis with voom consisted of only one contrast matrix of tumor vs luminal cells. This was visualized in Figure 3 in the form of a mean-difference plot. This plot showcases the minimal amount of DE genes found in the data set; the red points show up-regulated genes and the blue points show down-regulated genes in tumor cells

compared to the luminal cells. These are the genes analyzed in Figures 4 and 5, which we discussed earlier. The interactive plot by Glimma allowed me to analyze each of the DE genes and formulate my strategy for the remainder of the analysis. However, if we had made our filtration step less stringent, there may have been a higher amount of DE genes to analyze with the disadvantage of knowing these genes could include false information.

Overall, this analysis conducted showcased an interesting amount of findings that would be subject to future directional studies. The specific up- and down- regulated genes could be analyzed to determine their exact role in tumorigenesis and how each gene is affected by the loss of BRCA1. If I was going to develop this work into a larger project, or as my PhD thesis, I would first start by using a larger number of mice and extract both luminal and tumor samples from each; I believe this would be a more accurate comparison for the difference in gene expression. I would also take the time to determine the exact arguments and criteria for each step of the analysis to better my methodology and create a thorough conclusion. Based on the enrichment analysis, I would also go further and analyze the pathways of each of the differentially expressed genes and correlate them to the functionality of BRCA1 and how its loss affected each pathway. Even the results I was able to create would be a good starting point in analyzing pathway association by using gage or pathfindR. However, the outcome I was able to achieve did aid in answering my original question; the DE genes are functionally associated with the loss of BRCA1 and the formation of tumorigenesis.

Reflection: The script showcases the struggle I had while using this data set found on GEO. I believe it is usually difficult to understand the workings of someone else's research when the publicly available data is not properly annotated. This proves how important proper data labeling is for both yourself, and the others viewing your work. Each sample from this data had its own GEO page associated with it but all the information about the samples was duplicated to say the same for every sample. The only difference I was able to find was the difference in mouse name based on SC number and whether each mouse donated luminal or tumor samples. Working through the vignettes was very difficult because of this; I doubted my every step and was concerned that the results were incorrect. I would re-check and redo every step multiple times just to make sure I was entering the right arguments. Analyzing each outcome was also difficult as my plots varied from the vignette example at almost every step. This was definitely a steep learning curve for me and I hope I can learn from my mistakes to be better for the next time. I look forward to your comments and suggestions.

Acknowledgments

1. Amanda Meuser
    - Helped me with understanding R Markdown functions such as adding figure labels.
2. Eden Blanchard
    - Gave me the idea to add an extra color bar under my dendrogram showcasing the different groups.
3. Jacqueline May
    - Support through answering my questions and feedback on my opinions for the assignment.
4. Dr. Sarah Adamowicz -
    - Discussed the nature of my data set and helped me come to a conclusion for my next steps.
    - RNA-seq script and interpretation helped me understand the limma Bioconductor vignette and my results.
5. Study group: Emily Maier, Nykole Crevits and Patricia Balbon
    - Continuous encouragement and support while working through this assignment, and the term as a whole.

Vignettes

1. Carlson. M., Aboyoun, P., Pagès, H., Falcon, S., Morgan, M. (2021). Making and Utilizing TxDb Objects. https://bioconductor.org/packages/release/bioc/vignettes/GenomicFeatures/inst/doc/GenomicFeatures.pdf

2. Law, C. W., Alhamdoosh, M., Su, S., Dong, X., Tian, L., Smyth, G. K., & Ritchie, M. E. (2016). RNA-seq analysis is easy as 1-2-3 with limma, Glimma and edgeR. F1000Research, 5, ISCB Comm J-1408. https://master.bioconductor.org/packages/release/workflows/vignettes/RNAseq123/inst/doc/limmaWorkflow.html

3. Law, C. W., Zeglinski, K., Dong, X., Alhamdoosh, M., Smyth, G. K., & Ritchie, M. E. (2020). A guide to creating design matrices for gene expression experiments. F1000Research, 9, 1444. https://bioconductor.org/packages/release/workflows/vignettes/RNAseq123/inst/doc/designmatrices.html

4. Young, M., Wakefield, M., Smyth, G.n & Oshlack, A. (2012). goseq: Gene Ontology testing for RNA-seq datasets. https://bioconductor.org/packages/devel/bioc/vignettes/goseq/inst/doc/goseq.pdf

5. https://bioconductor.org/packages/release/data/annotation/manuals/TxDb.Mmusculus.UCSC.mm39.refGene/man/TxDb.Mmusculus.UCSC.mm39.refGene.pdf (No authors for the vignette)

Websites

1. Jrakru56, 2016. R: make the text bold for the legend and the xlab & ylab parameters in a plot? https://stackoverflow.com/questions/53177810/r-make-the-text-bold-for-the-legend-and-the-xlab-ylab-parameters-in-a-plot

2. Mi H, Huang X, Muruganujan A, Tang H, Mills C, Kang D, Thomas PD. PANTHER version 14: more genomes, a new PANTHER GO-slim and improvements in enrichment analysis tools. Nucleic Acids Res. Jan 2019;47(D1):D419-D426. http://geneontology.org/docs/go-enrichment-analysis/

3. user9112767, 2018. How to hold figure position with figure caption in pdf output of knitr? https://stackoverflow.com/questions/29696172/how-to-hold-figure-position-with-figure-caption-in-pdf-output-of-knitr

4. Young, Matthew D, Matthew J Wakefield, Gordon K Smyth, and Alicia Oshlack. 2010. "Gene Ontology Analysis for Rna-Seq: Accounting for Selection Bias." Genome Biology 11: R14. https://bioinformatics-core-shared-training.github.io/cruk-summer-school-2020/RNAseq/extended_html/06_Gene_set_testing.html

5. https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE148569 GEO link for obtaining the data set

References

1. Evans, C., Hardin, J., & Stoebel, D. M. (2018). Selecting between-sample RNA-Seq normalization methods from the perspective of their assumptions. Briefings in bioinformatics, 19(5), 776–792.

2. Giavarina D. (2015). Understanding Bland Altman analysis. Biochemia medica, 25(2), 141–151.

3. Hayashizaki Y. (2003). RIKEN mouse genome encyclopedia. Mechanisms of ageing and development, 124(1), 93–102.

4. Hein, S. M., Haricharan, S., Johnston, A. N., Toneff, M. J., Reddy, J. P., Dong, J., Bu, W., & Li, Y. (2016). Luminal epithelial cells within the mammary gland can produce basal cells upon oncogenic stress. Oncogene, 35(11), 1461–1467.

5. Jhanwar-Uniyal M. BRCA1 in cancer, cell cycle and genomic stability. Front Biosci. 2003 Sep 1;8:s1107-17.

6. Law, C. W., Chen, Y., Shi, W., & Smyth, G. K. (2014). voom: Precision weights unlock linear model analysis tools for RNA-seq read counts. Genome biology, 15(2), R29.

7. Seyednasrollah F, Laiho A, Elo LL. Comparison of software packages for detecting differential expression in RNA-seq studies. Brief Bioinform. 2015 Jan;16(1):59-70.

8. Sun H, Zeng J, Miao Z, Lei KC, Huang C, Hu L, Su SM, Chan UI, Miao K, Zhang X, Zhang A, Guo S, Chen S, Meng Y, Deng M, Hao W, Lei H, Lin Y, Yang Z, Tang D, Wong KH, Zhang XD, Xu X, Deng CX. Dissecting the heterogeneity and tumorigenesis of BRCA1 deficient mammary tumors via single cell RNA sequencing. Theranostics 2021; 11(20):9967-9987.

9. Yoshida K, Miki Y. Role of BRCA1 and BRCA2 as regulators of DNA repair, transcription, and cell cycle in response to DNA damage. Cancer Sci. 2004 Nov;95(11):866-71.